Adam Harvey Explains Viola-Jones Face Detection

Adam Harvey is an artist, designer, and programmer in New York. His work explores the intersection of privacy, computer vision, and fashion. His ongoing project, CV Dazzle, is an attempt to create hair styles, makeup, and clothing that thwart face recognition software. CV Dazzle began as Adam's thesis at NYU's Interactive Telecommunications Program and has required him to deeply investigate the workings of the Viola-Jones algorithm that underlies most common face detection software.

In this interview I asked Adam about his work and then had him explain the Viola-Jones algorithm to me in as much detail as possible. Viola-Jones is an important building block for many other face-related computer vision techniques including Jason Saragih's FaceTracker.

Transcript

Hi, Adam, how's it going?

Good.

So, I want to start off by talking about your work and how you came to be interested in face tracking. So, will you start off by telling us about the anti-paparazzi clutch?

Sure. One of my first projects at ITP was the anti-paparazzi clutch. This is one of the first prototypes, which is not that pretty. It's kind of nice.

I started working on this project because I used to work as a photographer and when I moved to New York some of the jobs that I got were photographing parties. It wasn't paparazzi work, but it was along the lines a little bit. And they required me to go up to people that didn't always want to have their photo taken, take their photo and then I would have to deal with that.

I thought well it's such a predatory art to be so aggressive in your art making.

Mentioned Links

- Adam Harvey
- CV Dazzle
- NYU's Interactive Telecommunications Program
- Jason Saragih's FaceTracker
- Cobrasnake
- lastnightsparty.com
- CV Dazzle
- Pia Vivas
- Processing
- JavaCVPro
- Viola-Jones algorithm
- Integral Image
- AdaBoost
- a good introduction to AdaBoost by its authors
- Principal Component Analysis
- PittPatt
- face.com
- Ohtsuka Masakazu's library
- here
- the Learning

Processing book

And that was congruent with this author Susan Sontag who wrote a lot about photography in the 1970s and the implications and the aggression involved in it.

Photography was my art tool, the camera, and I was really interested in how we were using digital cameras. So when I came out of school in 2004, that was when we were going through the big shift from film to digital. At that time I was working in a newspaper, so we switched from all the film cameras to all the digital cameras. What that meant for us as shooters was that we shot at least twice as many photos. And, of course, all these photos have to end up somewhere. And it's great if you are shooting parties to have all these extra outtakes because at parties candid photos are the ones that are great.

So, what I noticed happening and what lead me to this project was you have all these goofy candid photos that you wouldn't want to show, but now they're excess, so it's free to put them up online, go ahead and put them up there.

I think some people would make a business around this. There were a few photographers that really became popular around that time. Cobrasnake who people loved to hate, lastnightsparty.com and then a whole slew more after that.

I thought, oh man if I go out do something fun, that's maybe something I don't want to look back on in 10 years it wouldn't be fun anymore to see that photo online. So I thought what could I do that's involved with photography, still within the art of photography, but to change that image.

So I worked on this project and without knowing too much about technology or circuits or anything like that I was able to cram all the circuitry inside a purse enough to get the effect of overexposing the photo. And I was motivated to work on that, so I kept learning more about electronics and technology.

So the purse detects when a photo is being taken and turns on lights?

Oh, yeah. the way it works is by sensing when a flash is taken, just like a slave flash. And when the light is bright enough here it triggers all these [*indicating bright lights on the front of the bag*]. Inside the bag there is batteries and that's what powers the LEDs.

Cool. And then your thesis work at ITP, CV Dazzle, sounds like it grewout of a similar sense of privacy in photography?

Yeah, exactly. Going back to party photos online, you have the same problem, but now it's amplified because you can use really cheap computer labor to search through all those photos and mine identities. And then you have that same 10 year problem, but now it's now even easier to find somebody's face. I don't happen to be a party animal, but when it comes to privacy I think you want to be able to have fun and not worry about that stuff. So, I was really motivated by this aggression that's part of the image and how people use that as a piece of your privacy. It's your data also.

The aggression meaning connecting the image to your identity in that way.

Yeah, owning something that would have been a part of you. How do you share that? How do you deal with those problems? I came up with CV Dazzle, which is a way to style yourself when you go out. Again, it grew out of party makeup. It was inspired by these tribal decorations from Papua New Guinean cultures, as well as the BoomBox scene in London and combining a tribal look with something that's a little more futuristic. Actually, the hairstylist that I worked with

was inspired by Star Trek. So, you can see the first image that I did, the first design, the haircut is inspired by Spocks's hair. [laughs]

Like the chopped…

Yeah. That was Pia Vivas, the hairstylist.

In order to do that, in order to frustrate that kind of face detection, you had to learn about the details of howit works. Did you have experience with that before? Had you worked with face detection before that?

Yeah. I came into that project in 2010 with a barely workable knowledge of how to implement a face detection application in Processing. Processing is already one step removed from face detection. There's a group in France who have a website where you can get the Open CV library for processing (JavaCVPro). Then you link processing to that library. But it's so obscure that you have no idea what's going on. The cool thing is you can get it up and running in minutes. It's amazing the first time you do it. It's a cool experience.

What was the first thing you saw? Do you remember that experience? The first thing you sawon the screen, when seeing the face detection program run for the first time?

Yeah. I just did it of myself with my MacBook. You just try to move around to see if it's really working, just to validate it.

So, you're seeing the square around the face?

Yeah. I guess the conventional thing you do to identify the face is always this square. It's a square because it's actually a square that you're looking for. If you go then inside the algorithm, what you find is the Viola-Jones algorithm. The Viola-Jones algorithm works by looking for features. All the features are rectangles. Inside the rectangle are smaller rectangles and then you have pixels. The rectangle that makes up the face in Viola-Jones is very small.

It's funny when you learn about it. If you learn about any kind of face detection or face tracking, you learn that the face is really small. It's a small square in someone's head, which is just this little space here [*indicating his face*]. All of this other stuff is not even important. It makes you look at peoples' faces in a new, weird way. [laughs]

Rather than the kind of circle with a smiley face that we do when we're kids.

Yeah, right. For CV Dazzle I would try to understand what was happening in this algorithm and look at ways to expose the vulnerabilities of the Viola-Jones algorithm. Maybe I can show you something from here.

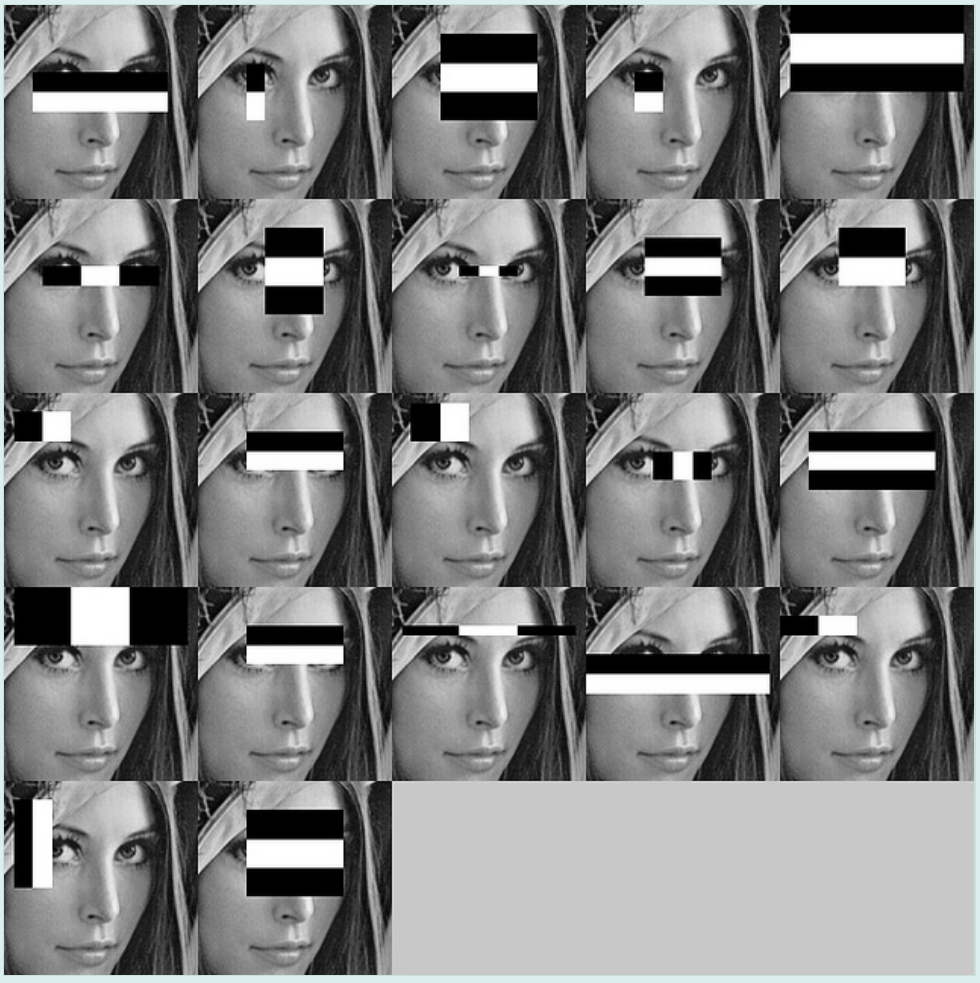
Yeah. Let's do that.

When I got further into the Viola-Jones algorithm and the OpenCV package, what I found are these files [*indicating a series of files named haar_cascade.xml and other similar things*]. And you get to choose these when you implement most any face detection in Processing or OpenFrameworks. You see these weird names like "Haar cascade", "frontal face", "alt tree", and there are a couple of them. These are very fine-tuned profiles for what makes up a

face. Those are four that are very common. People have made a lot of other variations of those.

You can see some of them I think on my screen here. There's one that makes up an eye [*called haardcascade_eye.xml*]. You can get one for a full body [*haarcascade_fullbody.xml*], left eye, right eye, glasses, mouth, nose, basically anything. There's even one I found for a clock. You can locate a clock in an image.

If you open up the XML file it's this really hard to read syntax. In the XML file it has a node with sub-nodes and if you go all the way into the last node, you see these numbers like 3, 8, 12, 4, -1, and these things define the shape of these things called Haar features. So the Haar features are basically black and white rectangles. And the black and white rectangles can be visualized. They actually are black and white.



An example of an early stage in the Haar cascade. Each black and white patch represents a feature that the algorithm hunts for in the image.

And the way they work in the algorithm is that you take the white rectangle and subtract the black rectangle, and then you ask, "Is that difference greater or less than the threshold?", and if it is or it isn't then you add it to a sum. And then you tally that sum and if the sum is within another threshold then it says, "OK that was what I was looking for, or not."

And we can look at that in code. I wrote a Processing sketch early on in this project to help me try to understand what's going on because these files are pretty hard to understand. So, what I did was look at the file and I learned there are stages to the cascade file. "Cascade file" — the name implies that it has a cascading structure. So in order for it to be efficient enough that you can use it

doing realtime video, its a very efficient algorithm and it's the one that led to the explosive growth of face detection everywhere across devices. It's very lightweight. And very efficient. But its weakness is that it can't do profiles very well. So if you are using this version of face detection that's one thing to keep in mind, that's it is mostly targeted towards frontal faces.

Could you bring that image back up for a second? Will you explain again? We're seeing the dark and light rectangles on top of the face, and as you mentioned, we're looking for a match between those rectangles and the image of the face at different scales. Would you say just a little bit more about that?

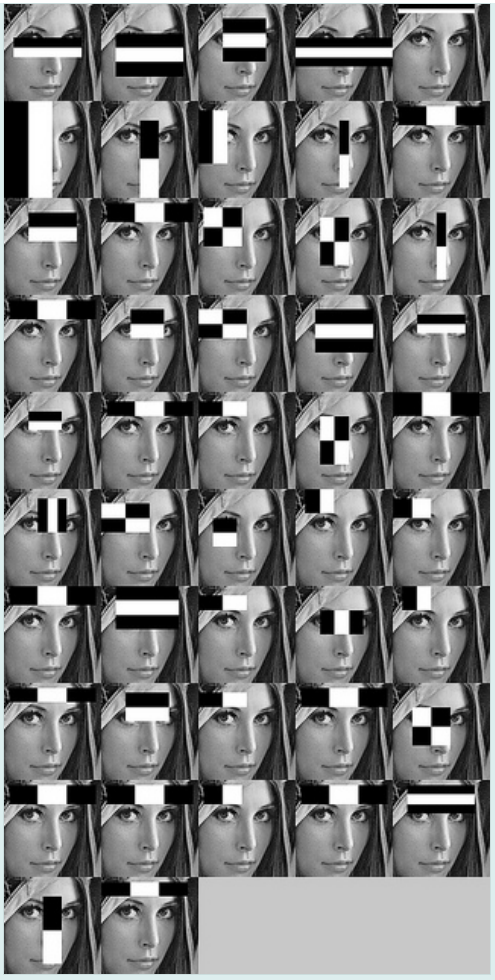
Yeah. I'll show you some more. This is stage zero [*indicating a Haar cascade image like the one above but with only three pictures*.] They typically have like 20 to 30 different stages. The first stage is a very coarse scan of the image. I'll talk about the scan in a second. But you can see in the first one here that there's a black rectangle and a white one. And if you compare that to the face below it you can understand that the white is the highlight of the cheeks and the black is the darkness of the eyes, the shadow. And then if you look at the second one, there's a lighter spot in the middle of the forehead and dark on the edges, which is where the hair would be. And then again on the third one the same thing as the second, but a little bit lower.

So you're comparing each of those rectangular patterns with the pixels in the face underneath to see howmuch that particular part of the image matches that pattern?

Well, what you do is called, I think, the Integral Sum [*the Integral Image or Summed Area Table*] of the part of that image. All you do is sum the black and white pixels. So all images are converted to black and white: grayscale, 0 to 255. You take these values and you create a sum of all the pixels in that area. So then you're comparing the sum of the grayscale values of the white to the sum of the grayscale values of the black region. And that's pretty efficient and easy to do in an image by calculating those sums. And Viola-Jones are the ones who came up with that method and it is very efficient. To really understand ityou do have to read their paper to get the formulas. It's easier to explain with formulas.

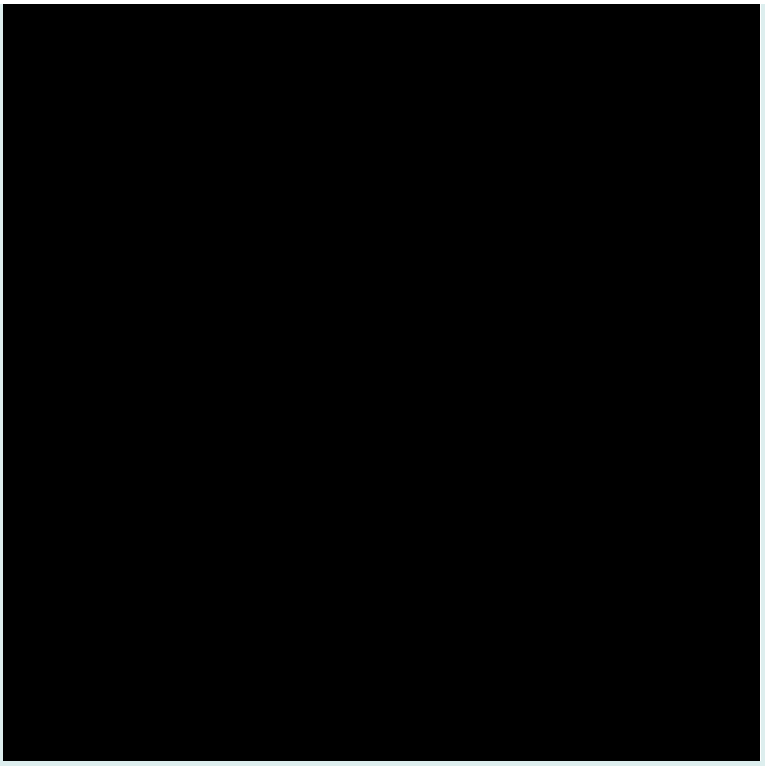
So then you can look at it as: "Is the sum of the white minus the sum of the black within a certain threshold?" And if it is then it'll either pass or fail the stage. What makes this algorithm efficient is that it can move past areas of an image that are not a face, rejecting them, and then focus on areas of the image that are more like a face. So if stage 0 is okay, then it goes to stage 2. Stage 2 gets a little more detailed, stage three is a harder test to pass, stage four is even harder.

You can see as you go further into the cascade these get increasingly complex and larger [*showing additional Haar cascade images as above but with ever more patterns of black and white rectangles that have to match regions of the face*], and they also take more time to compute and the thresholds are smaller. So by the time you get to the last stage you've got a hundred or more of these rectangles to compute. At that point it's probably a face. It's very close. And you can see here it's just a giant one.



An illustration of the later stage Haar cascade where many more patterns of black and white rectangles need to match the candidate image.

A big problem with face detection is false positives. If you get an area that looks like a face how do you know if it is? Because there are a lot of things that look like a face, just the shadows. Sometimes it happens on your t-shirt, these contours. What the algorithm does next is it looks at — and I'll pull up a demo here — overlapping regions. Maybe I'll pull up a different demo that works, we'll see [laughs]. How about I just pull up a video?



This is a visualization of all those cascades you just saw. I made a lot of videos. They're kind of abstract, but you can see all the different shapes and there are sometimes a thousand of them altogether. And when you overlay them all on top of an image you get something that looks like this here.

What you see is a red square moving across the image. That's called the scan window. Typically it starts small then grows larger so it will move very fast when it's large, it'll move very slow when it's small. The whole thing is slowed down to incredibly slow speeds here so you can look at it. Typically it would take a millisecond to scan an image like this.

What's happening is it's really fast looking at all these regions computing the difference. It's a little bit hard to see on the camera maybe but if it passes all of the stages you'll see a red square drop over that part of the image, I'll jump ahead. You can see here there are a couple of overlapping squares. And what I'm getting at here is even if it does pass in one area you do get a square that is

pretty close to being a face, but to avoid all the false positives the Viola-Jones algorithm then computes all these overlapping squares and there must be a minimum of…you can adjust it, but, say, 3. So if there are 3 overlapping squares in one area then chances are it's a face. These profiles have been tuned to get a really high accuracy. With a combination of a good face profile and the overlapping squares it's over 99% accurate. But again it's really made for frontal face positions.

A bit of interesting trivia for fun: this image which you always see associated with face detection and OpenCV is an image from a Swedish Playboy magazine. And you have that because you have a lot of people spending late hours working on this. I guess somebody did it as a joke and it's become tradition to use this model's image. Lena, that's the model's name.

So I just want to go over some of the parts. I think that was a good overview and I want to just go over some of the parts of the algorithm a little bit more just to make sure I understand howthey work. So, those cascade files that you have, that have that data, that comes from a training process that happens beforehand and the data is a summary of that training process. Can we talk a little bit more about the training?

Yeah. Training is something that I'm not an expert on but it's part of the OpenCV package and there's a script that takes a group of positives and a group of negatives and you label all the faces in the positives and then it runs a learning algorithm that uses something called AdaBoost [*a good introduction to AdaBoost by its authors* — ed.] which accelerates it. So it's a really large computational task to figure out all the sub-spaces because what these profiles have is rectangles that comprise a 24 by 24 space. Some of them are different but 24 by 24 is the most common. And if you think about a grayscale image that's 24 by 24 pixels, that's an astronomically huge number of possibilities within that space. And I could never compute it on here. So in order to accelerate that AdaBoost will steer left and right depending on if it's closer to the face on this side of a test pattern. So that algorithm, I've heard it even takes a day or so to run if you have large subset, maybe longer.

I have this folder that has a bunch of other cascades in it. And some of them are just crap. They don't work very well at all. And then if you go back to the frontal face you're like, 'wow this is actually pretty good'. So the clock one is good if you have a perfectly circular perfect clock but anything outside of that it doesn't pick up. So having good data is really important and something that came out of building these profiles is that it's really hard to build something for everybody, especially with occlusions and glasses and beards and all that stuff. And there were many other attempts and other algorithms: PCA, Principal Component Analysis, is one of them. And, of course now there are algorithms that are much better than this one and they can account for the different angles and different illumination. But the problem that this solves of the frontal face detection, it's really good and researchers at universities right now still use OpenCV, now 2.3, the latest one, but it's still using this. This is a rock solid face detection thing. The newer ones like iPhoto, Google, they acquired PittPatt, facebook, face.com… they're all using ones that are really efficient at these different angles of faces.

Do you knowthe names of some of those algorithms?

No. I think some of them may combine several different algorithms. PCA and Viola-Jones are the two that I'm most familiar with. So, it'd be really interesting to train your own profile and I think that would make for a cool other video because it's really useful in making your own applications if you can track

something custom.

How do you go about that? The cascades normally come with you OpenCV library so if you wanted to make one of those. Where do you go about doing that?

You could get a book to start with. This is something that's beyond me. I've never trained my own, but I would like to. It'd definitely be a rewarding experience if it turned out. But how would you go about doing that? There are instructions and I think the reason it hasn't become that popular is that it's just hard to get a really good profile and a large enough data set with quality data.

So let's go into a little bit more detail about the algorithm because I'm not sure I completely understand. I think I have the overall picture but I'm not sure I completely understand it. So, they're comparing those black and white patterns to a pre-computed average of each rectangular region of the image, that's stored efficiently at different scales. Is that right?

Well, the way the cascade of the profile is structured is XML files, and I'll open another one.

I guess, the question I'm trying to ask is: how do you tell when one particular one of those Haar features matches a part of the image?

If a Haar feature matches that part of the image. There are as I said maybe a thousand or so within a full profile. So if we go back to one of these visuals. For example, this is a good one to start with [*pulls up the first stage in the Haar cascade with just three face images overlaid with black and white rectangles*]. For example there's a rectangular region. So what do you do in your code is look at each pixel value, load all the pixels into array for this image, and you look at only the grayscale, color doesn't matter at all. And then within these rectangles you take the sum of every pixel that's in here. So you can't see because it's behind the black, but you take the sum of all these little pixels and average them together. Its the sum of everything in there. and then that is combined with all the other black areas in this and the black area is subtracted from the white area then you have your leftover value. These values are compared to thresholds in here. And then in here [*indicating a node in the XML file*] you can see a floating point number and in this case it's -0.0315. So if you take the average of all the white minus the average of all the black you're left with a value and if that's with in -0.03 then good it passed that one and then it goes on.

It's comparing contrast basically.

Yeah. And then what happens is if it passed or if it didn't pass it gets a value. There are two more values in each of these features. So in this XML you get the coordinates, the size of the feature, and the value if it fails or if it passes, and the threshold. So, going over that again. It says 'Is the sum within the threshold? Yes. Add this value'. Now each stage has a different number of tests. There are three here [*indicating the first stage of the Haar cascade*]. There are nine here [*indicating a later stage of the cascade*]. And then it just grows [*showing a later stage with even more tests*]. And we'll look at this one. What would happen in this stage is it would take the sum again of all the values at the end of the feature, so may be its get a little convolute here. But at the end of each stage you're left with a value of all the passed feature and failed features and if that value is within another threshold then you stop or go on to the next stage. So, that's the one reason why it's really a efficient algorithm, cause it just ignores all

the stuff that not going to be a face.

So you stop if it fails.

Yeah. Sometimes it will stop right at the first stage because it just a flat square it has no contrast.

So, you basically looking for a signature contrast pattern that we think might be one particular part of the face. Or you're just doing a quick test to see if that's there. And then at each stage you add more requirements that it match that set of contrast patterns in more specific spaces and you evaluate each one at a time and then the total amount it matches all those has to be within some threshold.

AYeah, more specific and finer thresholds. So also as you get all the way down to the top stage, the 25th stage, then you're dealing with really small differences in thresholds at that point and then you are looking at the difference of whether it's just a little bit darker here or a little bit lighter here [*indicating parts of his cheek*]. Whereas in the beginning it's like, 'is it a solid color or is there a little bit of contrast in in the face'.

So parts of the images with no contrast are eliminated very quickly?

Right. And once you learn all this information about the way it works you start to think in reverse and think, 'well of course if you had a flat shield over your face', it'd work pretty well [*to prevent face detection*], but it'd look pretty silly.

Yeah, doesn't make for the best makeup.

And I really wanted to try and outsmart the way this thing was working. So I got more and more into it and I found somebody had made a version of this library for ActionScript (Ohtsuka Masakazu's library, some English details here) and I was pretty familiar with ActionScript, so then I took that library and rebuilt it in Java. What you have in Processing or Java, is a link to the framework of OpenCV. The framework is all C++ files or C. This version in ActionScript is all ActionScript code. So if you know ActionScript then you can look at everything in there. You can look at everything. It's *Open* CV, but if you don't know C, it's harder. So I took that library and then rebuilt it in Java in Eclipse. What this does is give me more transparency into the way that it's working or that it's failing. You can see some the classes that make it up. There is a Haar feature for a two-rectangle feature. There is another one for a three-rectangle feature. Then, with this class, you can load up an image and get really detailed information about everything that is going on in the process. With this variant of OpenCV face detection you can see where it's failing in the image, you can see when it fails. If it's at the last stage, which means that it's just barely failing. And you can also see if it fails really early, that means that, for this project, it's a good pattern.

So that gave you more feedback about howgood of a job the makeup or hairstyle was doing at fooling the algorithm?

Yeah. And then I did a human learning trial and error until I became fluent through trial and error in which areas were affecting it more and less. When you get into it more, you can take these XML files. And it took me a long time to understand these. I emailed a bunch of random people on the web for explanations because there's not a lot of commenting in the OpenCV code about understanding the way it works. It's built in a way that makes sense if you just want to get it up and running and working and not really mess with the efficiency of it because all that

stuff has been taken care of. But sometimes you want to use it for something else. When you get an out of the box solution even things like OpenCV, which are pretty low level, what do you do then when you want to modify that? It becomes a whole other project. Viola-Jones has been around for such a long time, and it's still really popular. I think that's kind of cool. It's a rock solid face detection algorithm.

So does your Java implementation that is also producing those visualizations we looked at, those movies, and stuff like that? You're using that to do those?

Yeah.

What else are you looking at nowgoing forward with this project? Is that Java implementation public or going to be?

Yeah. It will be. It's almost useless if it's just me working on it. Maybe I'll have to do is maybe switch over to a more efficient language like C++ and OpenFrameworks and there's a lot of research and cool stuff coming out in there also. I think if I made something that exposed OpenCV, "Open-OpenCV", then people could start making branches of this and working on the same problem if it's interesting to them. Hopefully open source it. And there's some more interesting stuff going on with genetic algorithms. An interesting part of this project is that it wouldn't really work to just take a machine learning algorithm. and compute all of the patterns that are not faces because they would look horrible. So you have to have some kind of human guidance or assisted learning. And generic algorithms are great for that. There is a great example from Daniel Shiffman. I think it's in the Learning Processing book. He made this face evolver. So you can start with these geometric abstractions and then you can guide it towards the face. and that's the same idea for this project is that you can guide a design towards something that you would wear. So if you are a guy you can guide it towards a really cool design in your style. And your style is an important part of this too and I think that working with a lot of code stuff it's very easy to make a tool or a utilitarian type thing, that's kind of void of style. Generic algorithms are neat in that way because they allow for that, for stopping it in different parts of the process.

So you're thinking of producing or moving towards accoutrement that people could wear and use to actually beat face tracking in that way besides just make up?

Yeah. Hats, collars, double flip up collars maybe. I think it's a great time for fashion to incorporate some features that are more relevant to today. Like the original Levi's became successful because there're really a durable jean for miners to wear, but today whose mining? So what features can we add to jeans or clothing that are more relevant to our needs? If privacy is one of your needs, then we can certainly design something into clothing that would help with that. So another project that I started working on is this "off-pocket", which is taking EMF material, electric shielding material, it's metal-impregnated fabric and you get different metals for different effects. There is nickel, some silver-plated. Different metals have different effects as do different weaves. So the idea there is to make a pocket that is more suitable towards protecting your data, so it basically turns your phone off when you put it in and prevents data leakage in your pants.

Would that stop RFID? Scanning of RFID tags in your passports, too?

Yeah, sure. They're a little different frequency, but you can do that also.

So you'd need a different weave of mesh?

Yeah, or you with just a little more work you could probably design a universal shielding pocket.

Are there any other, like what next things you want to put on Open CV., like [inaudible]

Cool. That was a really interesting survey of your next projects you're thinking about. What next things do you want to learn about OpenCV? What did you come out of this project being intrigued about wanting to learn more of in computer vision?

I was just starting it completely overwhelmed. I thought OpenCV was a good place to start. I thought it was the lowest common denominator. And there are all these other companies pouring money into it, from the government to large companies like Google and Facebook. There is so much research money, there must be some amazing things that I don't know about. So how do I even get into that field? And I did email a bunch of different companies and ask them and most of them were pretty friendly about talking about which algorithms they used for their product. Because I think once you get into the research community it's pretty much open who's doing what and you can tell. Although this project has lead me into some interesting new fields. I hadn't considered much biometrics because my gateway into all this stuff was photography. And now photography is relevant to a certain extent, but there are all these other biometrics. You have gate detection, you have iris detection, fingerprint, handwriting analysis. So all these other things factor in, but it's a new field. It's rich and unexplored still so there are a lot of opportunities to explore new stuff and I if anyone is coming into an art field with this kind of background, an engineering or science of biometrics, they have an advantage for making something really cool.

So is there anything else that you can think of about Haar detection or the project? For example why is it called "Haar detection" when the paper is Viola-Jones?

Well, I just guess, I think there was somebody named, "Haar". [*laughs*]

I guess those are Haar features that they're naming it for?

Yeah. And I think you see those in other alogirhtms too, not just the Viola-Jones. Anything else? Well cool project I am looking forward to seeing more interviews.

Thanks. Yeah this was great. It really helped me understand. Thanks a lot.

Thank you.